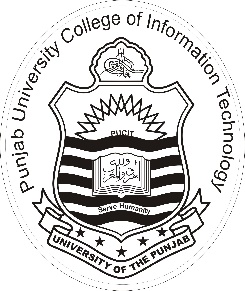
****

PUCIT

Punjab University College of Information Technology

**FemStore Application**

**(Application for Buying and Selling Products**

**Clone like Gilgit app- buy/ sell products in Pakistan)**

**Second Deliverable for Object Oriented Approach**

**Supervisor: Sir Wasim Ahmed**

**Project Team Members:**

**BCSF18M508- AROOJ ALI**

**BCSF18M534- IQRA PARVEEN**

**TABLE OF CONTENT:**

[1 Introduction 2](#_Toc148483291)

[1.1 Usecase descriptions 3](#_Toc148483292)

[1.2 Use Case Diagram (Refined) 4](#_Toc148483293)

[1.3 Domain Model 6](#_Toc148483293)

[1.4 Sequence Diagram 9](#_Toc148483295)

[1.5 Collaboration Diagram 10](#_Toc148483300)

[1.6 Operation Contracts 10](#_Toc148483303)

[1.7 Design Class Diagram 16](#_Toc148483304)

[1.8 Data Model 18](#_Toc148483313)

**1 Introduction:**

This deliverable is all about the usecase modeling and software design. In the previous deliverable, all analysis of our system is completed(The scope is defined, application interface is designed , milestone description and roles and tasks are assigned and start and end dates of the project is finalized). So now we understand the current situation of the problem domain. Now we are ready to strive for a solution for the problem domain by using object-oriented approach. Following artifacts must be included in this deliverable.

1. Use case description
2. Use case diagram refined
3. Domain Model
4. Sequence Diagram
5. Collaboration Diagram
6. Operation Contracts
7. Design Class Diagram
8. Data Model

Now we discuss these artifacts one by one as follows:

**1.1 Usecase Description**

**Use Case UC1: FemStore App**

**Primary Actors: Buyer, Seller**

**Stakeholders and Interests**

-**Seller:** Wants to sell items by providing necessary details , also manage and update personal information if wants.

- **Customer/Buyer**: Wants purchase and fast service with minimal effort. , also manage and update personal information if wants.

**Preconditions:** Buyers and sellers are identified and authenticated.

**Success Guarantee (Postconditions):** A user/buyer enters in the home (for buyer to buy the product and for seller to sell the item ) can add their favourite product to the wishlist for buyer and for seller he/she can see their Ads that he/ she uploaded.

**Main Success Scenario (or Basic Flow):**

1. Seller/ Buyer arrives at FemStore App to sell/ purchase but before this he/she must have to register/login.

2. Seller post ad by providing product information.

3. Buyer browse the products and add the favorite products to the wishlist.

4. Buyer can do chats with the sellers in case if they have any question.

5. Seller see posted products can be shown in the my ads if he/she wants.

6. The buyer/ seller go to settings and make necessary changes if any.

**Extensions (or Alternative Flows):**

\*a. At any time, System fails:

To support recovery, ensure all events can be recovered from any step of the scenario.

1. if user forget password for login then he/she can the forget password then he/she reset the password and then can login.

**1.2 Use case Diagram (refined and updated):**

Analysis level use case diagram is a refined High level use case diagram and is actually the explanation of high level use case diagram. This explains each and every flow of functionality that our app exhibit . Two types of relationships are used in this diagram. Which are:

* Extend
* Include

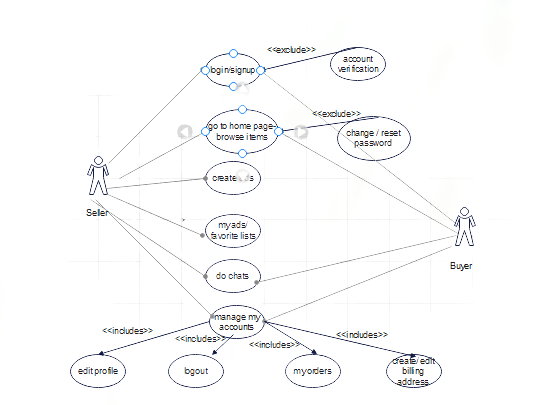
**Include Use Cases:**

**Signup** : Signup use case includes the verification without verify the user (Buyer/Seller) is not allowed and authenticated

**My Accounts**: this use case has includes the four farther major use case (Edit Profile, Uploads , Logout, My Orders)

**Exclude Use Cases:**

**Sign/Login:** Forget/ change password use case excluded from this use case as it extend the functionality but not the required functionality.

****

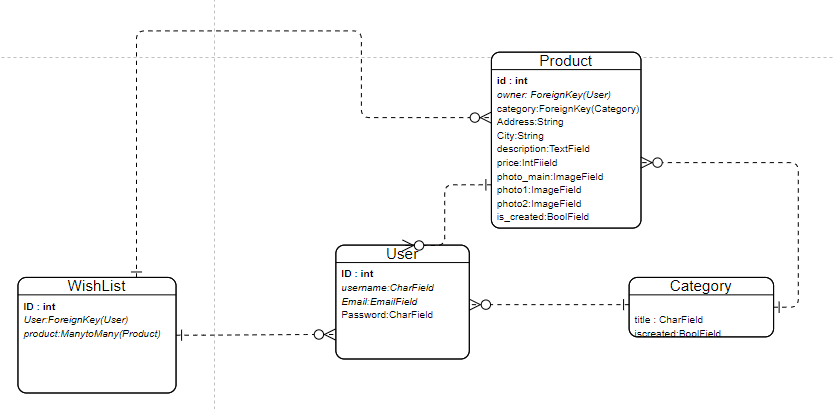
## 1.3 Domain Model

A domain model is an "incomplete" business model, in that it omits individual worker responsibilities. The point of domain modeling is to provide "the big picture" of the interrelationships among business entities in a complex organization. The domain model typically shows the major business entities, and the relationships among the entities

This domain model mainly provide our entities and their associations

**Major Entities:**

* Product
* User
* Category(Product or Cities of Pakistan to be searched for)
* WishList

****

**Domain Model Description:**

* **Association between Product and User:**

Here product and user has one or many relation, user may be sell at least one product and many and buyer can buy single or no or many product.

* **Association between Product and Category:**

Product and Category has one to many relation. Every Product must have category either product category like (Clothing, fashion, accessories etc) or by city (Lahore, ..).

* **Association between Category and User:**

User must search for at least one category product or by city category to browse or search the item or for seller to post ad , he/she must add by city category or product type category.

* **Association between Wish List and User:**

User and wish list has zero or many relationship , user may add one or no product or many product of his/her own choice

* **Association between Product and Wish list:**

Product and Wish list has many to many relation , multiple or no product can be added in the wishlist**.**

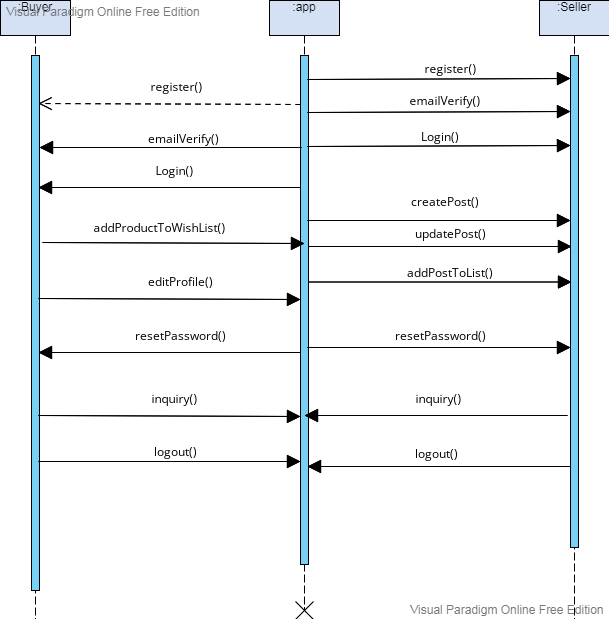
**1.4-SequenceDiagram:**

A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram

A Sequence diagram is two-dimensional in nature. On the horizontal axis, it shows the life of the object that it represents, while on the vertical axis, it shows the sequence of the creation or invocation of these objects.

**Class roles**

* **Buyer**
* **Seller**
* **App**

****

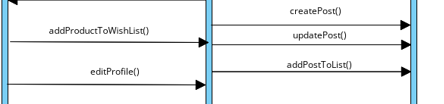
**Sequence Activities for login and Sign up:**

Sign up with email verification and login with credentials by buyer and seller to the app

****

**Sequence Activities for Home Page:**

Add post,view products, add products to wish list , update ads data are manage in this ares

****

**Chats in the Home page**

****

**Sequence Activities for My Accounts:**

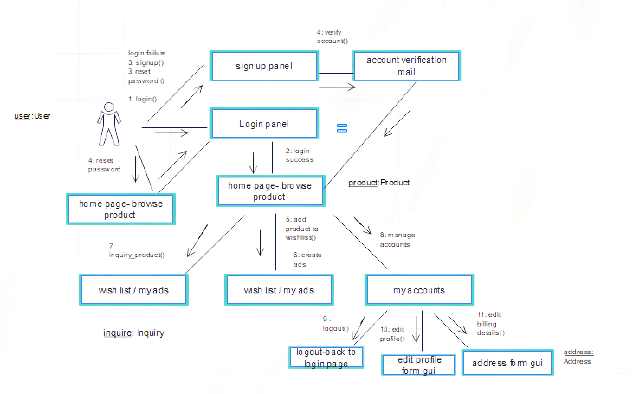
Logout from the app done by My Accounts

****

## 1.5 Collaboration Diagram

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaborations are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces

Collaboration Diagram for our project is as follows :

****

**It describes the basic flow the user enters the store and perform and manage relative activities**

**Major Actors:**

* **Seller**
* **Buyer**

**1.6 Operational Contract**

**Contract: sign up**

**Operation:** register()

**Cross References:** Use Cases: Femstore app

**Exception**: Empty fields entered

**Pre-conditions:** the user is first time visited the app.

**Post-conditions:** User successfully receive email for his/her account creation

**Contract: confirm\_register**

**Operation:** emailverify()

**Cross References:** Use Cases: Femstore app

**Exception**: may not get email

**Pre-conditions:** User has filled sign up form

**Post-conditions:** user successfully entered the login page

**Contract: reset\_password**

**Operation:** reset\_password()

**Cross References:** Use Cases: Femstore app

**Exception**: Empty fields entered

**Pre-conditions:** user must have done sign-up activity

**Post-conditions:** user password updated

**Contract: login**

**Operation:** login()

**Cross References:** Use Cases: Femstore app

**Exception**: Empty/Invalid fields entered

**Pre-conditions:** User has an account on this app

**Post-conditions:** User successfully ends to the home page

**Contract: create \_ad\_post**

**Operation:** create\_add()

**Cross References:** Use Cases: Femstore app

**Pre-conditions:** User account is created

**Exception**: may leave required fields

**Post-conditions:** user see their added product in my ads section

**Contract: ad\_post\_to\_list**

**Operation:** at\_post\_list()

**Cross References:** Use Cases: Femstore app

**Exception**: none

**Pre-conditions:** product created successfully

**Post-conditions:** my ad cart updated successfully

**Contract: update\_post**

**Operation:** update\_post()

**Cross References:** Use Cases: Femstore app

**Exception**: Empty fields or may not fill required fields

**Pre-conditions:** User already has that post

**Post-conditions:** User post also updated in my ads

**Contract: addproducts\_to\_list**

**Operation:** add\_to\_wishlist ()

**Cross References:** Use Cases: Femstore app

**Pre-conditions:** Product is successfully created

**Exception**: none

**Post-conditions:** wishlist updated successfuly

**Contract: About**

**Operation:** billing\_info()

**Cross References:** Use Cases: Femstore app

**Pre-conditions:** Customer has an account on this app

**Exception**: none

**Post-conditions:** Read all terms and conditions

**Contract: edit\_profile**

**Operation:** edit\_profile()

**Cross References:** Use Cases: Femstore app

**Exception**: Empty fields entered or may not filled required fields

**Pre-conditions:** User has an account on this app

**Post-conditions:** profile updated successfully

**Contract: logout**

**Operation:** logout()

**Cross References:** Use Cases: Femstore app

**Exception**: none

**Pre-conditions:** User has an account on this app

**Post-conditions:** User successfully return to login page

**Contract: inquirys**

**Operation:** inquirys()

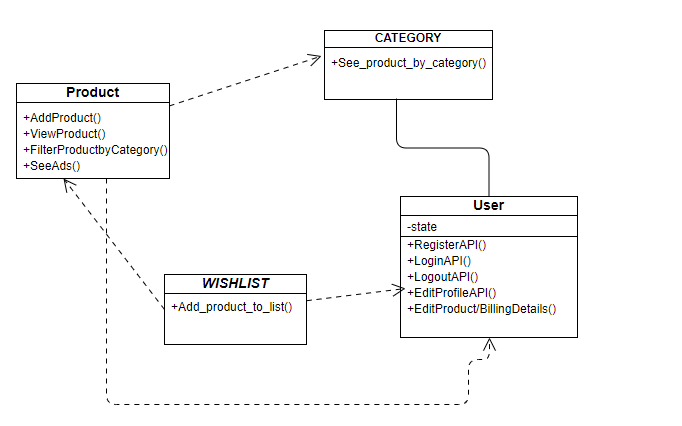
**Cross References:** Use Cases: Femstore app

**Exception**: none

**Pre-conditions:** Both buyer and seller information has already created

**Post-conditions:** buyer seller will successfully negoitating about the product

**1.7 Design Class Diagram**



**Define Operation Visibility**:

All member functions of the classes are private as the application system functionality based upon

REST API framework. So all member functions of the class in this respect is private only the apis /urls are sent to the front end using GET,POST,PUT methods to handle the backend functionality

**Dependency of classes:**

**Product class dependency:** On all the classes

**Product class User:** has no dependency

**Wishlist class dependency**: depend on Users and Product class

**Category class dependency:** has no dependency with any other class

**Define Class Operations:**

**Operations of Product Class:**

**Addproduct()🡪** API function that can create product of all required fields

**Methods allowed:**

* **GET**
* **POST**

**Viewproduct()🡪** API function that allow users to view products shown in the “homepage of app”

**Methods allowed:**

* **GET**

**FilterproductbyCategory()🡪** API function that can create function to filter/browse product by category to the buyer

**Methods allowed:**

* **GET**

**SeeAds()🡪** API function that can create see ads options for seller to see their ads crested of all required fields

Methods allowed:

* **GET**

**Operations of User Class:**

**RegisterAPI ()🡪** API function that can create user with all required fields

**Methods allowed:**

* **GET**
* **POST**

**EditProfileAPI ()🡪** API function through which user can update his/her profile **information**

**Methods allowed:**

* **GET**
* **POST**

**LoginAPI ()🡪** API function through which user can enter his /her credentials to enter into app for performing further activities.

**Methods allowed:**

* **GET**
* **POST**

**LogoutAPI ()🡪** API function through which user can logout and end his/her sessionfrom the app

**Methods allowed:**

* **GET**

**Operations of Category Class:**

**See\_product\_by\_categoryAPI ()🡪** API function through which user browse product by city name or product name fields

**Methods allowed:**

* **GET**

**Operations of Wish List Class:**

**Add\_product\_to\_listAPI ()🡪** API function through which user or buyer move their selected / desired products to wishlist

**Methods allowed:**

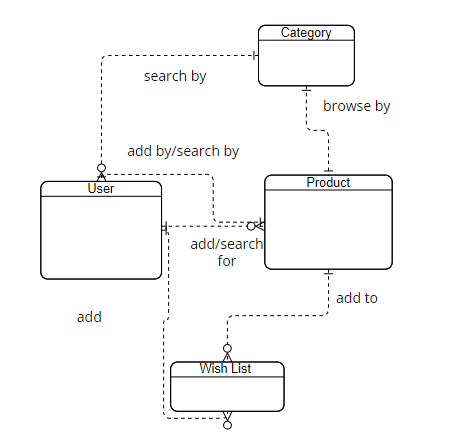
* **GET**

## 1.8 Data Model

The data model is a subset of the implementation model, which describes the logical and physical representation of persistent data in the system.

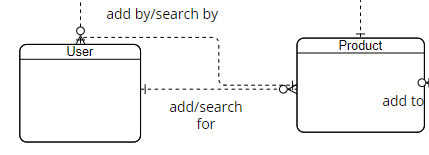
**Tables for the Data Model:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **User** | **Product** | **Category** | **Wishlist** |
| User |  | add/search | Search by | add |
| Product | add by/ search for |  | Browse by | Add to |
| Category | Search by | Browse by |  |  |
| Wish list | Add | Add to |  |  |

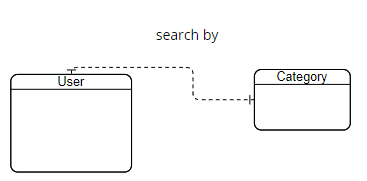


**Entities and their Relation separately:**

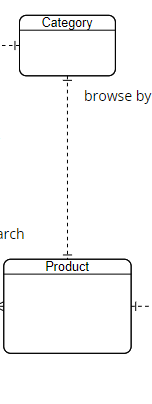
**Relation of Product with User( One to Many)**



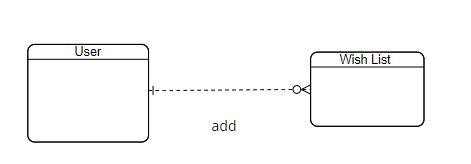
**Relation of User with Category( One to One)**



**Relation of Product with Category( One to One)**



**Relation of User with Wish list One to Many)**



**Relation of Product with Wish List ( One to Many)**

